

Web Cloud: Web-Based Cloud Storage for Secure Data Sharing Across Platforms

A Project Report Submitted To

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, ANANTAPUR

*In partial fulfillment of the requirements
for the award of the degree of*

**BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING**

Submitted by
IV B. Tech II Semester

K.MANIDHAR	20KH1A0543
K.NITHYA MALLESWAR	20KH1A0552
K.SREENIVASA PRASAD	20KH1A0548
G.DINO RAHUL RAJ	20KH1A0539
M. PAWAN SAI	20KH1A0557
L.GIRIDHAR	20KH1A0556

Under the Esteemed Guidance of

Mr. G. SREENIVASULU M. Tech.,(Ph.D.,)
ASSISTANT PROFESSOR & HoD



Department of Computer Science and Engineering

Sri Venkateswara College of Engineering

**(Affiliated to JNTUA, Anantapuramu and Approved by AICTE, New Delhi)
Balaji Nagar, Kadapa-516003.**

(2020-2024)

Web Cloud: Web-Based Cloud Storage for Secure Data Sharing Across Platforms

Project Report Submitted In Partial Fulfillment Of The Requirement For The Award
of the Degree of **B. Tech in Computer Science and Engineering**

K.MANIDHAR	20KH1A0543
K.NITHYA MALLESWAR	20KH1A0552
K. SREENIVASA PRASAD	20KH1A0548
G.DINO RAHUL RAJ	20KH1A0539
M.PAWAN SAI	20KH1A0557
L.GIRIDHAR	20KH1A0556

Under the Esteemed Guidance of

Mr. G. SREENIVASULU M. Tech.,(Ph.D.,)
ASSISTANT PROFESSOR & HoD



Department of Computer Science and Engineering

SV COLLEGE OF ENGINEERING

Approved by AICTE, New Delhi, Affiliated to JNTUA, Anantapuramu
Website: www.svck.edu.in



Sri Venkateswara College of Engineering

(Affiliated to JNTUA, Anantapuramu and Approved by AICTE, New Delhi)

Balaji Nagar, Kadapa-516003.

Department of Computer Science and Engineering

Certificate

This is to certify that the project work entitled

**Web Cloud: Web-Based Cloud Storage for Secure Data
Sharing Across Platforms**

is the bonafide work done by

K.MANIDHAR	20KH1A0543
K.NITHYA MALLESWAR	20KH1A0552
K. SREENIVASA PRASAD	20KH1A0548
G.DINO RAHUL RAJ	20KH1A0539
M.PAWAN SAI	20KH1A0557
L.GIRIDHAR	20KH1A0556

In the Department of Computer Science and Engineering, Sri Venkateswara College of Engineering, Kadapa is affiliated to JNTUA-Anantapur in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering during 2020-2024.

This work has been carried out under my guidance and supervision.

The results embodied in this Project report have not been submitted in any University or Organization for the award of any degree or diploma.

INTERNAL GUIDE

Mr. G. SREENIVASULU, M. Tech., (Ph.D.)
Assistant Professor & HoD

HEAD OF THE DEPARTMENT

Mr. G. SREENIVASULU, M. Tech., (Ph.D.)
Assistant Professor & HoD

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We are extremely thankful to our beloved chairman and establisher **Sri A. Gangi Reddy** for providing necessary infrastructure and resources for the accomplishment of our project at Sri Venkateswara College of engineering, Kadapa.

We are highly indebted to **Dr. R. Veera Sudarsana Reddy, Principal of Sri Venkateswara College of Engineering, Kadapa**, for his support during the tenure of the project.

We are very much obliged to our beloved **Mr. G. Sreenivasulu, Head of the Department of Computer Science & Engineering**, Sri Venkateswara College of Engineering for providing the opportunity to undertake this project and encouragement in completion of this project.

We here by wish to express deep sense of gratitude to **Mr. G. Sreenivasulu**, Department of Computer Science and Engineering, Sri Venkateswara College of Engineering for the esteemed guidance, moral support and invaluable advice provided by him for the success of the project.

We are also thankful to all the staff members of Computer Science and Engineering department who have co-operated in making our project a success. We would like to thank all our parents and friends who extended their help, encouragement and moral support either directly or indirectly in our project work.

Thanks for Your Valuable Guidance and kind support.

DECLARATION

We hereby declare that project report entitled **Web Cloud: Web-Based Cloud Storage for Secure Data Sharing Across Platforms** is a genuine project work carried out by us, in B.Tech (Computer Science and Engineering) degree course of **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR** and has not been submitted to any other courses or University for award of any degree by us.

Signature of the Student

1. K.Manidhar
2. K.Nithya Malleswar
3. K.Sreenivasa Prasad
4. G.Dino Rahul Raj
5. M. Pawan Sai
6. L.Giridhar

ABSTRACT

With more and more data moving to the cloud, privacy of user data have raised great concerns. Client-side encryption/decryption seems to be an attractive solution to protect data security, however, the existing solutions encountered three major challenges: low security due to encryption with low-entropy PIN, inconvenient data sharing with traditional encryption algorithms, and poor usability with dedicated software/plugins that require certain types of terminals. This work designs and implements WebCloud, a practical browser-side encryption solution, leveraging modern Web technologies. It solves all the above three problems while achieves several additional remarkable features: robust and immediate user revocation, fast data processing with offline encryption and outsourced decryption. Notably, our solution works on any device equipped with a Web user agent, including Web browsers, mobile and PC applications. We implement Web Cloud based on ownCloud for basic file management utility, and utilize Web Assembly and Web Cryptography API for complex cryptographic operations integration. Finally, comprehensive experiments are conducted with many well-known browsers, Android and PC applications, which indicates that WebCloud is cross-platform and efficient. As an interesting by-product, the design of WebCloud naturally embodies a dedicated and practical ciphertext-policy attribute-based key encapsulation mechanism (CP-AB-KEM) scheme, which can be useful in other applications.

LIST OF FIGURES

Fig. No.	Figure Name	Page No.
Figure 5.3.1	Use Case Diagram	13
Figure 5.3.2	Class Diagram	14
Figure 5.3.3	Sequence Diagram	15
Figure 5.4.4	Activity Diagram	16
Figure 5.3.5	Context Level Data Flow Diagram	18
Figure 6.1	Working of Java Program	20
Figure 6.2	Implementation of Java Virtual Machine	20
Figure 6.3	Program Running on the Java Platform	21

LIST OF TABLES

Table No.	Table Name	Page No.
Table 9.3	Test Case Results	37

LIST OF SCREENS

Screen No.	Screen Name	Page No.
Screen 11.1.	Home Page and Login Page	47
Screen 11.2.	Cloud Server Login	47
Screen 11.3.	Data Owner Login	48
Screen 11.4.	Data Owner Registration	48
Screen 11.5.	User Owner Login	49
Screen 11.6.	User Owner Registration	49
Screen 11.7.	Private Key Generator Login	50
Screen 11.8.	Private Key Generator Main	50

LIST OF ABBREVIATIONS

S.No.	Abbreviation	Full Form
1.	WWW	World Wide Web
2.	JSP	Java Server Pages
3.	GUI	Graphical User Interface
4.	UML	Unified Modeling Language
5.	HTML	Hyper Text Markup Language
6.	DFD	Data Flow Diagram
7.	SDLC	Software Development Life Cycle

CONTENTS

ABSTRACT	iii
LIST OF FIGURES	iv
LIST OF TABLES	v
LIST OF SCREENS	vi
LIST OF ABBREVIATIONS	vii
 CHAPTER	 Page No.
1. INTRODUCTION	1
1.1. ABOUT THE PROJECT	1
1.2. PROJECT DESCRIPTION	1
1.3. MODULES OF THE PROJECT	1
1.3.1. Data Owner Module	1
1.3.2. Data User Module	3
1.3.3. Cloud Server Module	3
2. SYSTEM ANALYSIS	4
2.1. DOMAIN ANALYSIS	4
2.2. REQUIREMENT ANALYSIS	4
2.2.1. Functional Requirements	4
2.2.2. Non-Functional Requirements	4
2.2.3. User Interfaces	5
2.2.4. Software Interface	5
2.2.5. Manpower Requirements	5
2.3. EXISTING SYSTEM	5
2.3.1. Disadvantages	5
2.4. PROPOSED SYSTEM	5
2.4.1. Advantages	6
3. FEASIBILITY STUDY	7
3.1. TECHNICAL FEASIBILITY	7
3.2. ECONOMICAL FEASIBILITY	7
3.3. OPERATIONAL FEASIBILITY	8
4. HARDWARE AND SOFTWARE REQUIREMENTS	9
4.1. HARDWARE REQUIREMENTS	9
4.2. SOFTWARE REQUIREMENTS	9
5. SYSTEM DESIGN	10

5.1. UML DIAGRAMS INTRODUCTION	10
5.2. SYSTEM DESIGN ASPECTS	10
5.2.1. Design of Code	10
5.2.2. Design of Input	11
5.2.3. Design of Output	11
5.2.4. Design of Control	11
5.3. UML DIAGRAMS	12
5.3.1. Use Case Diagram	13
5.3.2. Class Diagram	14
5.3.3. Sequence Diagram	15
5.3.4. Activity Diagram	16
5.3.5. Data Flow Diagram	17
6. OVERVIEW OF TECHNOLOGIES	19
6.1. JAVA TECHNOLOGY	19
6.2. THE JAVA PLATFORM	21
6.3. ODBC	23
6.4. JDBC	23
7. OVERVIEW OF DBMS	26
7.1. DATA ABSTRACTION	26
7.2. INSTANCES AND SCHEMAS	26
7.3. DATA MODELS	26
7.3.1. The Entity Relationship Model	26
7.3.2. Relational Model	27
7.4. DATABASE LANGUAGES	27
7.4.1. Data Definition Language	27
7.4.2. Data Manipulation Language	27
7.5. ORACLE	27
7.5.1. Introduction	27
7.5.2. Distinct Features of Oracle	28
7.5.3. Features of Oracle	28
8. IMPLEMENTATION	31
9. TESTING	33
9.1. SOFTWARE TESTING TECHNIQUES	33
9.1.1. Testing Objectives	33
9.1.2. Test case Design	33
9.2. SOFTWARE TESTING STRATEGIES	34

9.2.1. Unit Testing	34
9.2.2. Integration Testing	34
9.2.3. Validation Testing	35
9.2.4. System Testing	36
9.2.5. Security Testing	37
9.2.6. Performance Testing	37
9.3. TEST CASES	37
10. SOURCE CODE	38
11. OUTPUT SCREENS	47
12. CONCLUSION AND FUTURE ENHANCEMENTS	51
12.1. CONCLUSION	51
12.2. FUTURE ENHANCEMENTS	51
13. REFERENCES	52

1. INTRODUCTION

1.1. ABOUT THE PROJECT

With more and more data moving to the cloud, privacy of user data have raised great concerns. Client-side encryption/decryption seems to be an attractive solution to protect data security, however, the existing solutions encountered three major challenges: low security due to encryption with low-entropy PIN, inconvenient data sharing with traditional encryption algorithms, and poor usability with dedicated software/plugins that require certain types of terminals. This work designs and implements WebCloud, a practical browser-side encryption solution, leveraging modern Web technologies. It solves all the above three problems while achieves several additional remarkable features: robust and immediate user revocation, fast data processing with offline encryption and outsourced decryption. Notably, our solution works on any device equipped with a Web user agent, including Web browsers, mobile and PC applications. We implement WebCloud based on ownCloud for basic file management utility, and utilize WebAssembly and Web Cryptography API for complex cryptographic operations integration. Finally, comprehensive experiments are conducted with many well-known browsers, Android and PC applications, which indicates that WebCloud is cross-platform and efficient. As an interesting by-product, the design of WebCloud naturally embodies a dedicated and practical ciphertext-policy attribute-based key encapsulation mechanism (CP-AB-KEM) scheme, which can be useful in other applications..

1.2. PROJECT DESCRIPTION

The Objective focuses on designing and implementing a practical, secure and cross-platform public cloud storage system. The proposed solution, WebCloud, is a Web-based client-side encryption solution. Users encrypt and decrypt their data using Web agents, e.g., Web browsers.

1.3. MODULES OF THE PROJECT

1.3.1. Data Owner Module

In this module, the data owner uploads their data in the cloud server. For the security purpose the data owner encrypts the data file's blocks and then store in the

cloud. The data owner can check the replication of the file's blocks over Corresponding cloud server. The Data owner can have capable of manipulating the encrypted data file's blocks and the data owner can check the cloud data as well as the replication of the specific file's blocks and also he can create remote user with respect to registered cloud servers. The data owner also checks data integrity proof on which the block is modified by the attacker. The cloud computing paradigm has reformed the usage and management of the information technology infrastructure. Cloud computing is characterized by ondemand self-services, ubiquitous network accesses, resource pooling, elasticity, and measured services. The aforementioned characteristics of cloud computing make it a striking candidate for businesses, organizations, and individual users for adoption. However, the benefits of low-cost, negligible management (from a users perspective), and greater flexibility come with increased security concerns. Security is one of the most crucial aspects among those prohibiting the wide-spread adoption of cloud computing. Cloud security issues may stem due to the core technologies. Client-side encryption/decryption seems to be an attractive solution to protect data security, however, the existing solutions encountered three major challenges: low security due to encryption with low-entropy PIN, inconvenient data sharing with traditional encryption algorithms, and poor usability with dedicated software/plugins that require certain types of terminals. This work designs and implements WebCloud, a practical browser-side encryption solution, leveraging modern Web technologies. It solves all the above three problems while achieves several additional remarkable features: robust and immediate user revocation, fast data processing with offline encryption and outsourced decryption. Notably, our solution works on any device equipped with a Web user agent, including Web browsers, mobile and PC applications. We implement WebCloud based on ownCloud for basic file management utility, and utilize WebAssembly and Web Cryptography API for complex cryptographic operations integration. Finally, comprehensive experiments are conducted with many well-known browsers, Android and PC applications, which indicates that WebCloud is cross-platform and efficient. As an interesting by-product, the design of WebCloud naturally embodies a dedicated and practical ciphertext-policy attribute-based key encapsulation mechanism (CP-AB-KEM) scheme, which can be useful in other applications..

1.3.2. Data User Module

The data user can check the replication of the file's blocks over Corresponding cloud server. The Data owner can have capable of manipulating the encrypted data file's blocks and the data owner can check the cloud data as well as the replication of the specific file's blocks and also he can create remote user with respect to registered cloud servers. Data User operation are following they are User Registration Login Request for Keys Receive Keys File download

1.3.3. Cloud Server Module

The cloud service provider manages a cloud to provide data storage service. Data owners encrypt their data file's blocks and store them in the cloud for sharing with Remote User. To access the shared data file's blocks, data consumers download encrypted data file's blocks of their interest from the cloud and then decrypt them.

2. SYSTEM ANALYSIS

Analysis is a logical process. The objective of this phase is to determine exactly what must be done to solve the problem. Tools such as Class Diagrams, Sequence Diagrams, data flow diagrams and data dictionary are used in developing a logical model of system.

2.1. DOMAIN ANALYSIS

Domain analysis is the process by which a software engineer learns background information, which helps to understand the problem. The word ‘domain’ in the case means the general field of business or technology in which the customers expect to be using the software. For this project, personal experiences of the team members with competing software were observed to understand the domain.

2.2. REQUIREMENT ANALYSIS

A requirement is a relatively short and concise piece of information, expressed as a fact. It can be written as a sentence or can be expressed using some kind of diagram.

2.2.1. Functional Requirements

Functional requirements describe what the system should do. The functional requirements can be further categorized as follows:

1. What inputs the system should accept?
2. What outputs the system should produce?
3. What data the system must store?
4. What are the computations to be done?

The input design is the link between the information system and the user. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

1. What data should be given as input?
2. How the data should be arranged or coded?
3. The dialog to guide the operating personnel in providing input.
4. Methods for preparing input validations and steps to follow when error occur.

2.2.2. Non-Functional Requirements

Non-functional requirements are the constraints that must be adhered during development. They limit what resources can be used and set bounds on aspects of the software's quality.

2.2.3. User Interfaces

The User Interface is a GUI developed using Java.

2.2.4. Software Interfaces

The main processing is done in Java and console application.

2.2.5. Manpower Requirements

5 members can complete the project in 2 – 4 months if they work fulltime on it.

2.3. EXISTING SYSTEM

Meanwhile, there are researches in the literature having explored the idea of running cryptographic algorithms on Web browsers. [29] focused on using Identity-Based Cryptography for client side security in Web applications and presented a JavaScript implementation of their scheme. They selected Combined Public Key cryptosystem as the encryption scheme to avoid complex computations involved in bilinear pairing and elliptic curve.

2.3.1. Disadvantages

- Comparatively poor security,
- Coarse-grained access control, inflexible and inefficient file sharing, and
- Poor usability. The first two are easy to see and we now elaborate the usability issue. Typically, users use different terminals to upload files, including desktop, Web and mobile applications

2.4. PROPOSED SYSTEM

The proposed system focuses on designing and implementing a practical, secure and cross-platform public cloud storage system. The proposed solution, WebCloud, is a Web-based client-side encryption solution. Users encrypt and decrypt their data using Web agents, e.g., Web browsers. The proposed system implemented Multi-Factor Authenticated Key Exchange which gives more security and safe. The uniform design, rigorous analysis and efficient implementation of WebCloud, in particular, it

simultaneously achieves the Web techniques and cryptographic algorithms. WebCloud involves of a key management mechanism, a dedicated attribute based encryption scheme and a high-speed implementation.

2.4.1. Advantages

- The experiments show greatly reduce the overhead on the client side, which only introduces a minimal additional cost on the server side.
- Such an approach is beneficial to implement a realistic data sharing security scheme on mobile devices.
- The results also show better performance compared to the existing ABE based access control schemes over ciphertext.
- The storage overhead needed for access control is very small compared to data files.

3. FEASIBILITY STUDY

All projects are feasible when provided with unlimited resources and infinite time. Unfortunately, the development of computer-based system or product is more likely plagued by a scarcity of resources and difficult delivery dates. It is both necessary and prudent to evaluate the feasibility of a project at the earliest possible time. Months or years of effort, thousands or millions of dollars, and untold professional embarrassment can be averted if an ill-conceived system is recognized early in the definition phase.

Feasibility and risk analysis are related in many ways. If project risk is great the feasibility of producing quality software is reduced. During product engineering, however, we concentrate our attention on four primary areas of interest.

3.1. TECHNICAL FEASIBILITY

This application is going to be used in an Internet environment called www (World Wide Web). So, it is necessary to use a technology that is capable of providing the networking facility to the application. This application is also able to work on distributed environment. Application is developed with Java Technology. One major advantage in application is platform neutral. We can deploy and use it in any operating system.

GUI is developed using HTML to capture the information from the customer. HTML is used to display the content on the browser. It uses TCP/IP protocol. It is an interpreted language. It is very easy to develop a page/document using HTML. Some RAD (Rapid Application Development) tools are provided to quickly design/develop our application. So many objects such as button, text fields, and text area etc are provided to capture the information from the customer.

3.2. ECONOMICAL FEASIBILITY

The economical issues usually arise during the economical feasibility stage are whether the system will be used if it is developed and implemented, whether the financial benefits are equal or exceeds the costs. The cost for developing the project will include cost of full system investigation, cost of hardware and software for

the class of being considered, the benefits in the form of reduced costs or fewer costly errors.

The project is economically feasible if it is developed and installed. It reduces the work load. Keep the class of application in the view, the cost of hardware and software is considered to be economically feasible.

3.3. OPERATIONAL FEASIBILITY

In our application front end is developed using GUI. So it is very easy to the customer to enter the necessary information. But customer must have some knowledge on using web applications before going to use our application.

4. HARDWARE AND SOFTWARE REQUIREMENTS

4.1. HARDWARE REQUIREMENTS

Processor	:	Core 2 Duo/ Core I3
Speed	:	Minimum 4.1 GHZ Speed
RAM	:	4 GB
Hard Disk	:	500 GB

4.2. SOFTWARE REQUIREMENTS

Operating System	:	Windows XP/7/8/10
Coding Languages	:	JAVA,JSP(JAVA SERVER PAGES)
Database	:	MY SQL
IDE	:	Eclipse
Web Server	:	Tomcat 7.0

5. SYSTEM DESIGN

5.1. UML DIAGRAMS INTRODUCTION

The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic, semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective.

UML is specifically constructed through two different domains they are:

- UML Analysis modeling, this focuses on the user model and structural model views of the system.
- UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

5.2. SYSTEM DESIGN ASPECTS

Once the analysis stage is completed, the next stage is to determine in broad outline form how the problem might be solved. During system design, we are beginning to move from the logical to physical level.

System design involves architectural and detailed design of the system. Architectural design involves identifying software components, decomposing them into processing modules and conceptual data structures, and specifying the interconnections among components.

Detailed design is concerned with how to package processing modules and how to implement the processing algorithms, data structures and interconnections of standard algorithms, invention of new algorithms, and design of data representations and packaging of software products. Two kinds of approaches are available:

- Top down approach
- Bottom up approach

5.2.1. Design of Code

Since information systems projects are designed with space, time and cost saving in mind, coding methods in which conditions, words, ideas or control errors and speed the entire process. The purpose of the code is to facilitate the identification and

retrieval of the information. A code is an ordered collection of symbols designed to provide unique identification of an entity or an attribute.

5.2.2. Design of Input

Design of input involves the following decisions

- Input data
- Input medium
- The way data should be arranged or coded
- Validation needed to detect every step to follow when error occurs

The input controls provide ways to ensure that only authorized users access the system guarantee the valid transactions, validate the data for accuracy and determine whether any necessary data has been omitted. The primary input medium chosen is display. Screens have been developed for input of data using HTML. The validations for all important inputs are taken care of through various events using JSP control.

5.2.3. Design of Output

Design of output involves the following decisions

- Information to present
- Output medium
- Output layout

Output of this system is given in easily understandable, user-friendly manner, Layout of the output is decided through the discussions with the different users.

5.2.4. Design of Control

The system should offer the means of detecting and handling errors.

Input controls provides ways to

- Valid transactions are only acceptable
- Validates the accuracy of data
- Ensures that all mandatory data have been captured

All entities to the system will be validated. And updating of tables is allowed for only valid entries. Means have been provided to correct, if any by change incorrect entries have been entered into the system they can be edited.

5.3 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

5.3.1 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

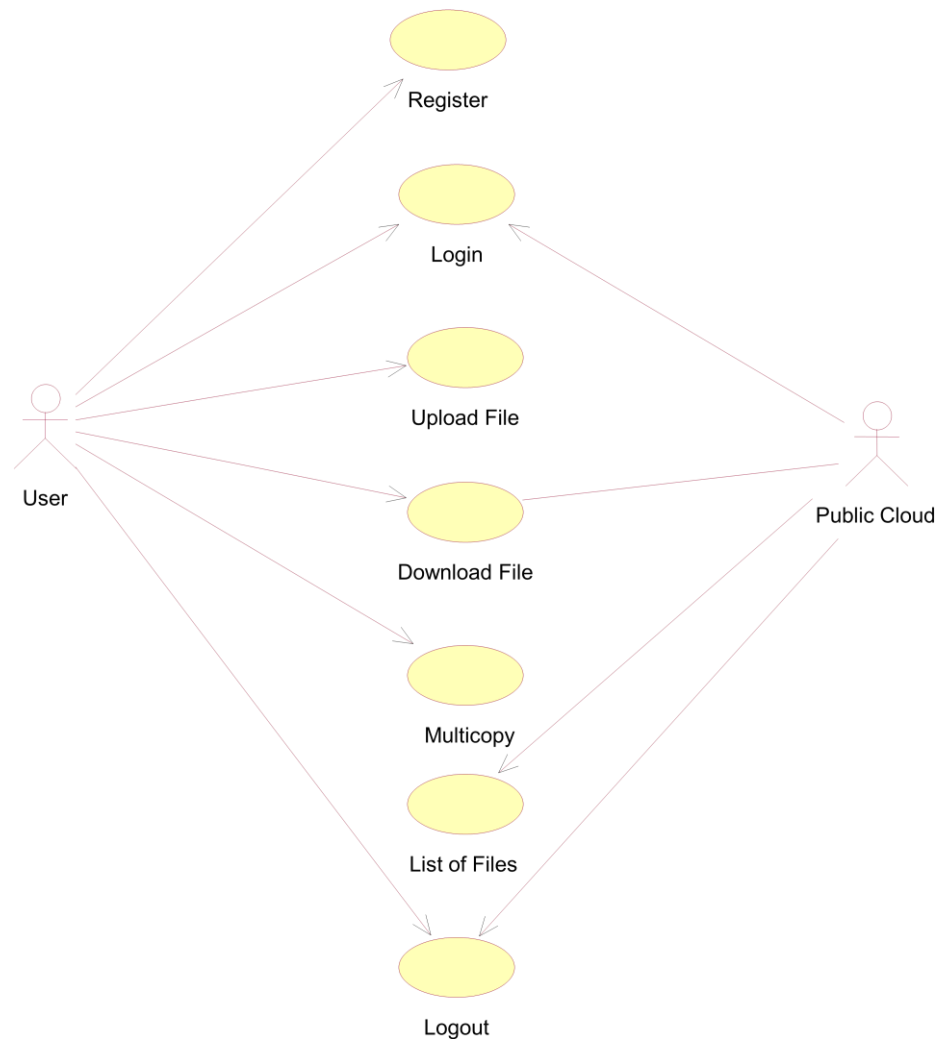


Fig 5.3.1. Use Case Diagram

5.3.2 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

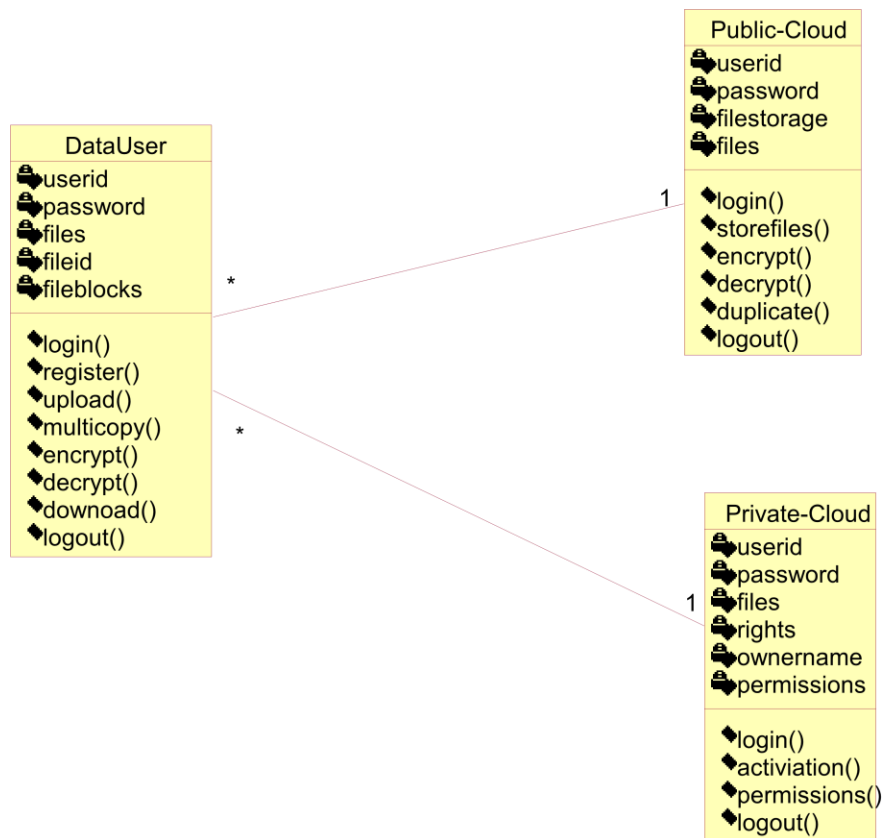


Fig 5.3.2. Class Diagram

5.3.3 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

SEQUENCE DIAGRAM FOR ADMIN:

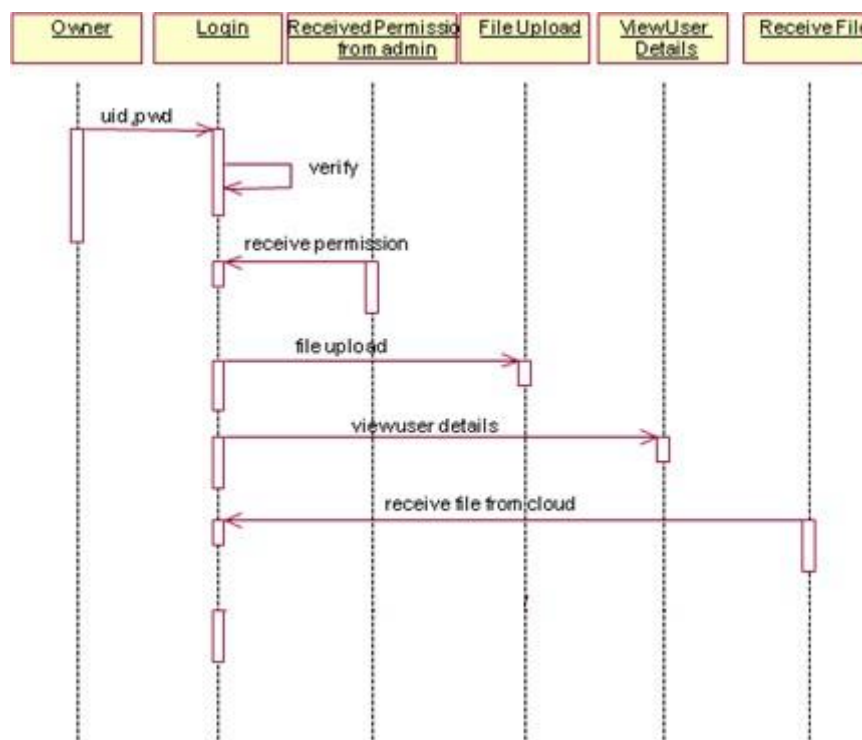


Fig5.3.3 Sequence Diagram For Admin

5.3.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

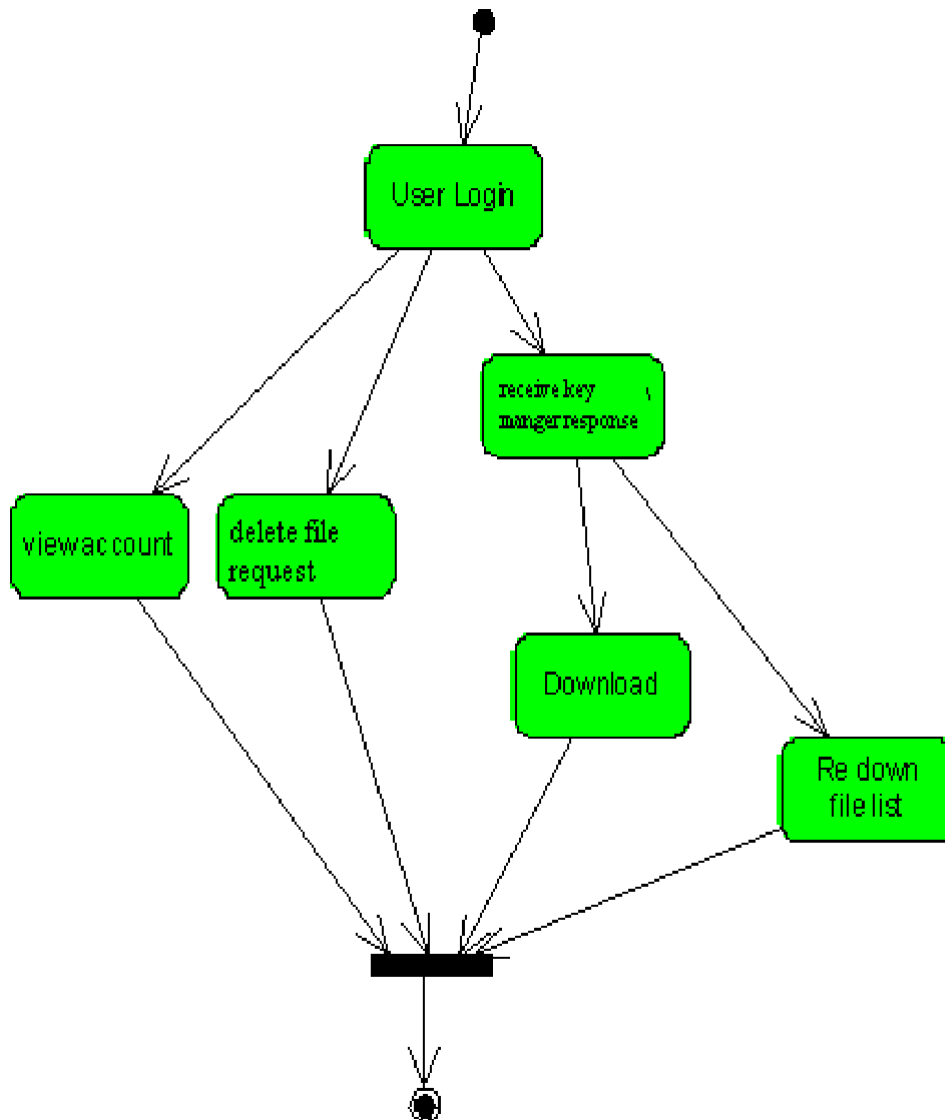
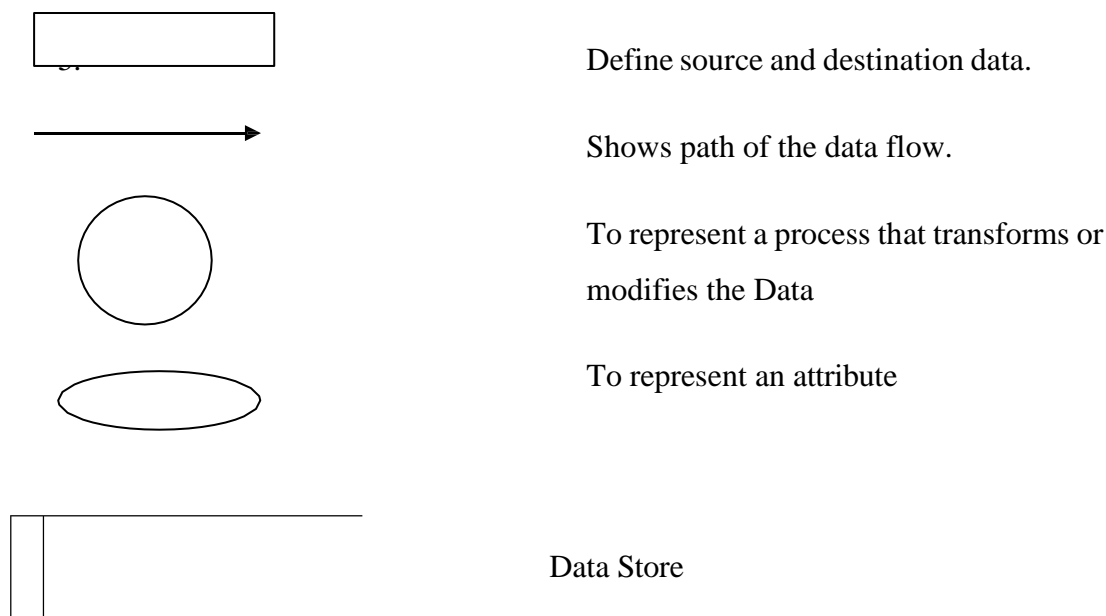


Fig 5.3.4 Activity Diagram

5.3.5 DATA FLOW DIAGRAM

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

DFD NOTATIONS



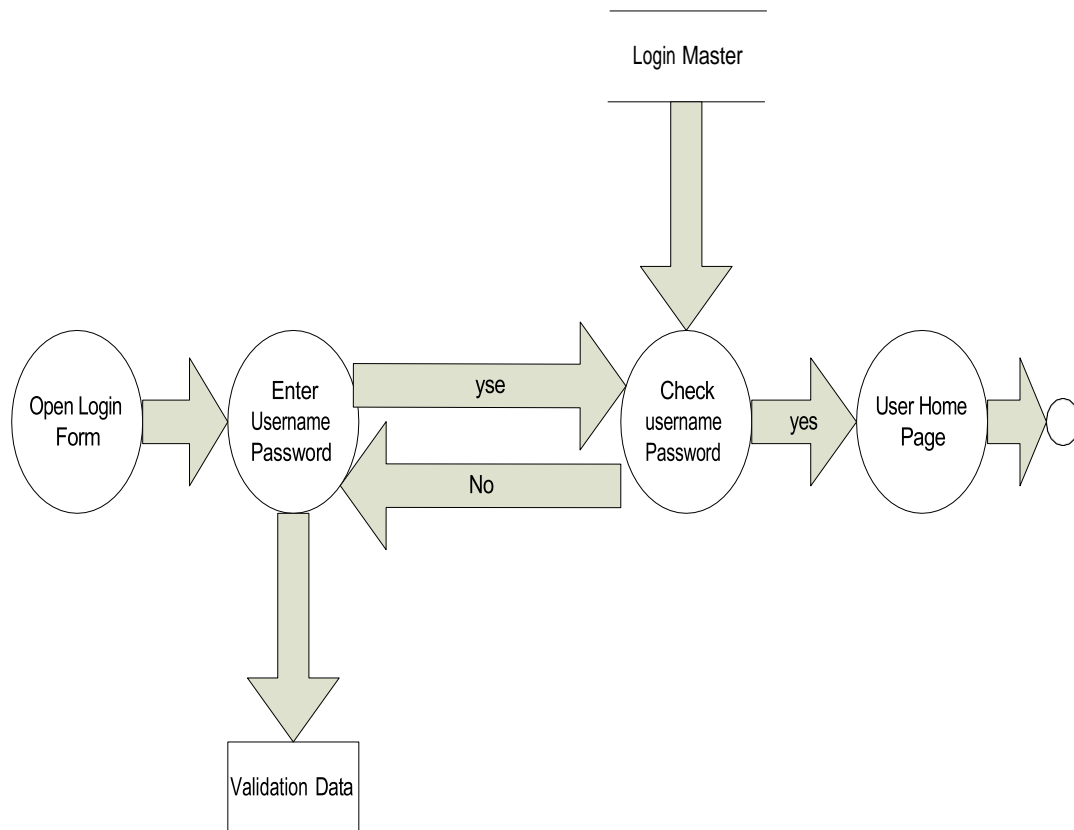


Fig 5.3.5. Context Level Data Flow Diagram

6. OVERVIEW OF TECHNOLOGIES

6.1. JAVA TECHNOLOGY

Java technology is both a programming language and a platform.

The Java Programming Language:

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

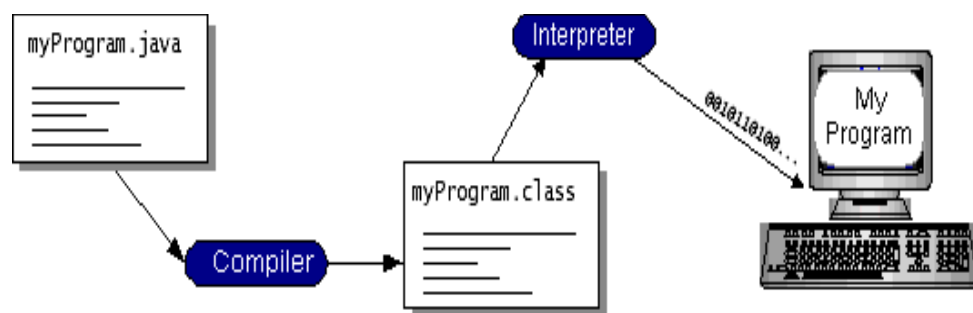


Fig 6.1. Working of Java Program

If we think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

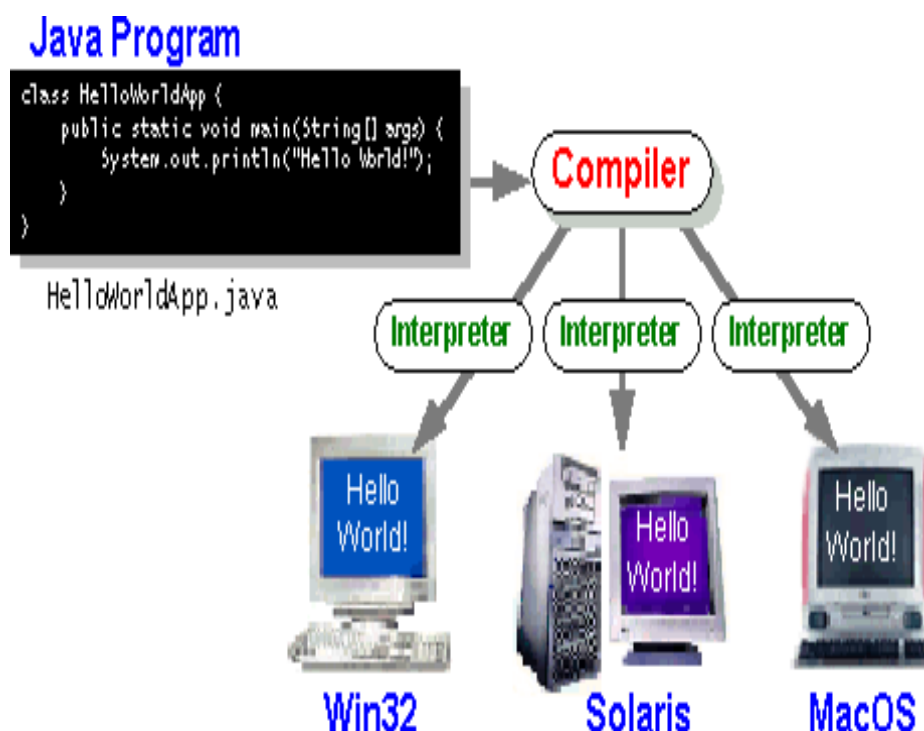


Fig 6.2. Implementation of Java Virtual Machine

6.2. THE JAVA PLATFORM

A platform is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The Java Virtual Machine (Java VM)
- The Java Application Programming Interface (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. In the next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

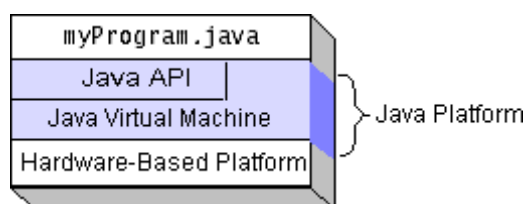


Fig 6.3. Program Running on the Java Platform

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

The most common types of programs written in the Java programming language are applets and applications. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a Servlet. A Servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, Servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The essentials:** Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets:** The set of conventions used by applets.
- **Networking:** URLs, TCP (Transmission Control Protocol), UDP (User Datagram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization:** Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security:** Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components:** Known as JavaBeansTM, can plug into existing component architectures.

- **Object serialization:** Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC):** Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.

6.3. ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a de facto standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

6.4. JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The design goals for JDBC are as follows:

SQL Level API:

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

SQL Conformance:

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

JDBC must be implemental on top of common database interfaces:

The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

Provide a Java interface that is consistent with the rest of the Java system:

Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

Keep it simple:

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

7. OVERVIEW OF DBMS

A Database Management System (DBMS) is a collection of interrelated data and set of programs to access those data. The primary goal of DBMS is to provide a way to store and retrieve database information.

7.1. DATA ABSTRACTION

Abstraction means to provide necessary information without considering the background details. There are three levels of abstraction for a DBMS.

- **Physical level:** It is lowest level of abstraction, which describes how the data was actually store on secondary device such as disks and tapes.
- **Logical level:** It is a second level of abstraction, which describes what data are stored in the database, and what relationships exist among those data. Database Administrators decide what data is to be kept in the database.
- **View level:** It is the highest level of abstraction, which describes only a part of the entire database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.

7.2. INSTANCES AND SCHEMAS

The collection of information stored in a database at a particular moment is called an instance. The overall design of a database is called a schema.

7.3. DATA MODELS

A Data Model is a collection of conceptual tools for describing data, data relationship, data semantic and consistency constraints. Various data models available are discussed below.

7.3.1. The Entity Relationship Model

E-R model is a data model used to describe the data involved in a real world enterprise. It describes the data in the form of entities and relationships. An entity is a ‘thing’ (or ‘object’) in the real world that can be easily distinguishable from other things. A relationship is an association among several entities.

7.3.2. Relational Model

The Relational Model uses a collection of tables to represent both data and the relationships among the data. Each table has multiple columns, and each column has a unique name.

7.4. DATABASE LANGUAGES

A database system provides data definition language and data manipulation language.

7.4.1. Data Definition Language

Data Definition Language (DDL) consists of a set of definitions used to specify data base schema. Execution of DDL statement results in a set of tables. These tables are stored in a specific area known as data dictionary or data directory. A data directory contains Meta data. Meta data is data about data.

7.4.2. Data Manipulation Language

Data manipulation is

- Retrieval of information stored in the database.
- Insertion of new information into the database.
- Deletion of information from the database.
- Modification of information stored in the database.

Data Manipulation Language (DML) is a language that enables users to access or manipulate data. There are basically two types.

- Procedural DMLs require a user to specify what data are needed and how to get those data.
- Declarative DMLs require user to specify what data needed without specifying how to get those data.

7.5. Oracle

7.5.1. Introduction

ORACLE is a relational database management system, which organizes data in the form of tables. ORACLE is one of many database servers based on RDBMS model, which manages a series of data that attends three specific things-data structures, data integrity and data manipulation. With ORACLE cooperative server technology we can realize the benefits of open, relational systems for all the applications. ORACLE makes efficient use of all systems resources, on all hardware architecture to deliver unmatched

performance, price performance and scalability. Any DBMS to be called as RDBMS has to satisfy Dr.E.F.Codd's rules.

7.5.2. Distinct Features of ORACLE

- **ORACLE is portable**

The ORACLE RDBMS is available on wide range of platforms ranging from PCs to super computers and as a multi user loadable module for Novel NetWare, if you develop application on system you can run the same application on other systems without any modifications.

- **ORACLE is compatible**

ORACLE commands can be used for communicating with IBM DB2 mainframe RDBMS that is different from ORACLE, that is ORACLE compatible with DB2. ORACLE RDBMS is a high performance fault tolerant DBMS, which is specially designed for online transaction processing and for handling large database applications.

- **Multithreaded server architecture**

ORACLE adaptable multithreaded server architecture delivers scalable high performance for very large number of users on all hardware architecture including symmetric multiprocessors (sumps) and loosely coupled multiprocessors. Performance is achieved by eliminating CPU, I/O, memory and operating system bottlenecks and by optimizing the Sql Server 2005, DBMS server code to eliminate all internal bottlenecks.

7.5.3 Features of ORACLE

Most popular RDBMS in the market because of its ease of use

- Client/server architecture.
- Ensuring data integrity and data security.
- Parallel processing support for speed up data entry and online transaction processing used for applications.
- DB procedures, functions and packages.

Dr.E.F.CODD's RULES

These rules are used for valuating a product to be called as relational database management systems. Out of 12 rules, a RDBMS product should satisfy at least 8 rules, +rule called rule 0 that must be satisfied.

RULE 0. FOUNDATION RULE

For any system that is to be advertised as, or claimed to be relational DBMS. That system should manage database within itself, without using an external language.

RULE 1. INFORMATION RULE

All information in relational database is represented at logical level in only one way as values in tables.

RULE 2. GUARANTEED ACCESS

Each and every data in a relational database is guaranteed to be logically accessible by using a combination of table name, primary key value and column name.

RULE 3. SYSTEMATIC TREATMENT OF NULL VALUES

Null values are supported for representing missing information and inapplicable information. They must be handled in systematic way, independent of data types.

RULE 4. DYNAMIC ONLINE CATALOG BASED RELATION MODEL

The database description is represented at the logical level in the same way as ordinary data so that authorized users can apply the same relational language to its interrogation as they do to the regular data.

RULE 5. COMPREHENSIVE DATA SUB LANGUAGE

A relational system may support several languages and various models of terminal use. However there must be one language whose statement can express all of the following Data Definitions, View Definitions, Data Manipulations, Integrity, Constraints, Authorization and transaction boundaries.

RULE 6. VIEW UPDATING

Any view that is theoretical can be updatable if changes can be made to the tables that effect the desired changes in the view.

RULE 7. HIGH LEVEL UPDATE, INSERT and DELETE

The capability of handling a base relational or derived relational as a single operand applies not only retrieval of data also to its insertion, updating, and deletion.

RULE 8. PHYSICAL DATA INDEPENDENCE

Application program and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access method.

RULE 9. LOGICAL DATA INDEPENDENCE

Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access methods.

RULE 10. INTEGRITY INDEPENDENCE

Integrity constraints specific to particular database must be definable in the relational data stored in the catalog, not in application program.

RULE 11. DISTRIBUTED INDEPENDENCE

Whether or not a system support database distribution, it must have a data sub-language that can support distributed databases without changing the application program.

RULE 12. NON SUB-VERSION

If a relational system has low level language, that low language cannot use to subversion or by pass the integrity rules and constraints expressed in the higher level relational language.

ORACLE SUPPORTS THE FOLLOWING CODD'S RULES:

- Rule 1: Information Rule (Representation of information)-YES.
- Rule 2: Guaranteed Access-YES.
- Rule 3: Systematic treatment of Null values-YES.
- Rule 4: Dynamic on-line catalog-based Relational Model-YES.
- Rule 5: Comprehensive data sub language-YES.
- Rule 6: View Updating-PARTIAL.
- Rule 7: High-level Update, Insert and Delete-YES.
- Rule 8: Physical data Independence-PARTIAL.
- Rule 9: Logical data Independence-PARTIAL.
- Rule 10: Integrity Independence-PARTIAL.
- Rule 11: Distributed Independence-YES.
- Rule 12: Non-subversion-YES.

8. IMPLEMENTATION

In the implementation phase software development is concerned with translating design specifications into source code. The primary goal of implementation is to write the source code internal documentation so that conformance of the code to its specification can be easily verified, and so that debugging, testing and modifications are erased. This goal is achieved by making the source code as clear and straightforward as possible. Simplicity, clarity and elegance are the hallmarks of good programs. Obscurity, cleverness and complexity are indications of inadequate design and misdirected thinking.

Source code clarity is enhanced by strutted techniques, by good coding style, by appropriate documents, by go internal comments, and by the features provided in the modern programming languages.

The main aim of structured coding is to adhere to single entry, single exit constructs in the majority of situations since it allows one to understand program behavior by reading the code from beginning to end. Bust strict adherence to this construct may cause problems it raises concerns for the time and space efficiency of the code. In some cases, single entry and single exit programs will require repeated code segments or repeated subroutines calls. In such cases, the usage of this construct would prevent premature loop exits and branching to exception handling code. So, in certain situations we violate this construct to acknowledge the realities of implementation although our intent is not encouraging poor coding style.

In computer programming, coding style is manifest in the patterns used by programmers to express a desired action or outcome good coding style can overcome the deficiencies of primitive programming languages, while poor style can defeat the intent of an excellent language. The goal of good coding style is to provide easily understood straightforward, elegant code.

Every good coding style performs the following Do's

- Introduce user-defined data types to model entities in the problem domain.
- Use a few standard, agreed-upon control statements.
- Hide data structures behind access functions.

- Use goto's in a disciplined way.
- Isolate machine dependencies in a few routines.
- Use indentation, parenthesis, blank lines and borders around comment blocks to enhance readability.
- Carefully examine the routines having fewer than 5 or more than 25 executable statements.

The following are the Don'ts of good coding style

- Avoid null then statements.
- Don't put nested loops very deeply.
- Carefully examine routines having more than five parameters.
- Don't use an identifier for multiple purposes.

Adherence implementation standards and guidelines by all programmers on a project results in a product of uniform quality. Standards were defined as those that can be checked by an automated tool. While determining adherence to a guideline requires human interpretation. A programming standard might specify items such as:

- The nested depth of the program constructs will not exceed five levels.
- The goto statements will not be used.
- Subroutines lengths will not exceed 30 Lines.

Implementation was performed with the following objectives

- Minimize the memory required.
- Maximize output readability or clarity.
- Maximize source text readability.
- To ease modification of the program.
- To facilitate formal verification of the program.
- To put the tested system into operation while holding costs, risks and user irritation to minimum.

Supporting documents for the implementation phase include all base-lined work products of the analysis and design phase.

9. TESTING

9.1. SOFTWARE TESTING TECHNIQUES

Software Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding, Testing presents an interesting anomaly for the software engineer.

9.1.1. Testing Objectives

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is one that has a probability of finding an as yet undiscovered error.
3. A successful test is one that uncovers an undiscovered error.

These above objectives imply a dramatic change in view port.

Testing cannot show the absence of defects, it can only show that software errors are present.

9.1.2. Test Case Design

Any engineering product can be tested in one of two ways:

White Box Testing

This testing is also called as glass box testing. In this testing, by knowing the specified function that a product has been designed to perform test can be conducted that demonstrates each function is fully operation at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis Path Testing

- Flow graph notation
- Cyclomatic Complexity

Deriving test cases Control Structure Testing

- Condition testing
- Data flow testing

- Loop testing

Black Box Testing

In this testing by knowing the internal operation of a product, tests can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing
- Graph matrices

9.2. SOFTWARE TESTING STRATEGIES

A Strategy for software testing integrates software test cases into a series of well planned steps that result in the successful construction of software. Software testing is a broader topic for what is referred to as Verification and Validation. Verification refers to the set of activities that ensure that the software correctly implements a specific function. Validation refers to the set of activities that ensure that the software that has been built is traceable to customer's requirements.

9.2.1. Unit Testing

Unit testing focuses verification effort on the smallest unit of software design that is the module. Using procedural design description as a guide, important control paths are tested to uncover errors within the boundaries of the module. The unit test is normally white box testing oriented and the step can be conducted in parallel for multiple modules.

9.2.2. Integration Testing

Integration testing is a systematic technique for constructing the program structure, while conducting test to uncover errors associated with the interface. The objective is to take unit tested methods and build a program structure that has been dictated by design.

- **Top-Down Integration**

Top down integrations is an incremental approach for construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main control program. Modules subordinate to the main program are incorporated in the structure either in the breath-first or depth-first manner.

- **Bottom-up Integration**

This method as the name suggests, begins construction and testing with atomic modules i.e., modules at the lowest level. Because the modules are integrated in the bottom up manner the processing required for the modules subordinate to a given level is always available and the need for stubs is eliminated.

- **Regression Testing**

In this contest of an integration test strategy, regression testing is the re execution of some subset of test that have already been conducted to ensure that changes have not propagate unintended side effects.

9.2.3. Validation Testing

At the end of integration testing software is completely assembled as a package. Validation testing is the next stage, which can be defined as successful when the software functions in the manner reasonably expected by the customer. Reasonable expectations are those defined in the software requirements specifications. Information contained in those sections form a basis for validation testing approach.

Reasonable expectation is defined in the software requirement specification – a document that describes all user-visible attributes of the software. The specification contains a section titled “Validation Criteria”. Information contained in that section forms the basis for a validation testing approach.

Validation Test Criteria

Software validation is achieved through a series of black-box tests that demonstrate conformity with requirement. A test plan outlines the classes of tests to be conducted, and a test procedure defines specific test cases that will be used in an attempt to uncover errors in conformity with requirements. Both the plan and procedure are designed to ensure that all functional requirements are satisfied, all performance requirements are achieved, documentation is correct and human-engineered; and other requirements are met.

After each validation test case has been conducted, one of two possible conditions exists: (1) The function or performance characteristics conform to specification and are accepted, or (2) a deviation from specification is uncovered and a deficiency list is created. Deviation or error discovered at this stage in a project can rarely be corrected prior to scheduled completion. It is often necessary to negotiate with the customer to establish a method for resolving deficiencies.

Configuration Review

An important element of the validation process is a configuration review. The intent of the review is to ensure that all elements of the software configuration have been properly developed, are catalogued, and have the necessary detail to support the maintenance phase of the software life cycle. The configuration review sometimes called an audit.

Alpha and Beta Testing

It is virtually impossible for a software developer to foresee how the customer will really use a program. Instructions for use may be misinterpreted. Strange combination of data may be regularly used; and output that seemed clear to the tester may be unintelligible to a user in the field.

When custom software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all requirements. Conducted by the end user rather than the system developer, an acceptance test can range from an informal “test drive” to a planned and systematically executed series of tests. In fact, acceptance testing can be conducted over a period of weeks or months, thereby uncovering cumulative errors that might degrade the system over time.

9.2.4. System Testing

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all system elements have been properly integrated to perform allocated functions.

9.2.5. Security Testing

Attempts to verify the protection mechanisms built into the system.

9.2.6. Performance Testing

This method is designed to test runtime performance of software within the context of an integrated system.

9.3. TEST CASES

Sl.NO.	Test Case	Excepted Result	Test Result
1	Enter valid name and password & click on login button	Software should display home page or user options page	Successful
2	Enter invalid details	Software should displays error page	Successful
3	Select Document or Text File	Selected document File and Text file should be visible on Page	Successful
4	Encrypt Document or Text	File will be encrypted	Successful
5	Decrypt document or text	Document will be decrypted	Successful
6	Report file	Reports all uploaded file	Successful
7	Verify File	File will be Verified	Successful
6	Recover File	File will be Recovered when Attacked	Successful

Table 9.3 Test Cases

10. SOURCE CODE

HOME.HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>HOME PAGE</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link href="css/style.css" rel="stylesheet" type="text/css" />

<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />

<script type="text/javascript" src="js/cufon-yui.js"></script>

<script type="text/javascript" src="js/droid_sans_400-
droid_sans_700.font.js"></script>

<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>

<script type="text/javascript" src="js/script.js"></script>

<script type="text/javascript" src="js/coin-slider.min.js"></script>

<style type="text/css">

<!--

.style2 {

    color: #FF0000;

    font-style: italic;

    font-weight: bold;

}
```

```
.style3 {color: #FF0000}

.style4 {

    font-size: 36px;

    color: #FF0000;

    font-weight: bold;

}

-->

</style>

</head>

<body>

<div class="main">

    <div class="header">

        <div class="header_resize">

            <div class="menu_nav">

<center><h2 style="color: yellow;" >Web Cloud Web-Based Cloud Storage for
Secure Data Sharing Across Platforms</center>

        <ul>

            <li class="active"><a href="index.html"><span>Home Page</span></a></li>

            <li><a href="DataUser.jsp"><span>Data Owner </span></a></li>

            <li><a href="EndUser.jsp"><span>End User </span></a></li>

            <li><a href="CloudServer.jsp"><span>Cloud Server </span></a></li>

        </ul>
```

```
</div>

<div class="slider">

    <div id="coin-slider"> <a href="#"> </a> <a href="#"> </a> <a href="#"> </a> </div>

    <div class="clr"></div>

</div>

<div class="clr"></div>

</div>

<div class="content">

    <div class="content_resize">

        <div class="mainbar">

            <div class="article">

                <p class="infopost">&nbsp;</p>

                <div class="clr"></div>

                <div class="img"></div>

                <div class="post_content">

                    <p align="justify" class="style2"></p>

                </div>

            </div>

        </div>

    </div>

</div>
```

```
<div class="clr"></div>

</div>

</div>

<div class="sidebar">

  <div class="searchform"></div>

  <div class="clr"></div>

  <div class="gadget">

    <h2 class="star"><span>Home </span> Menu</h2>

    <div class="clr"></div>

    <ul class="sb_menu">

      <li><strong><a href="index.html">Home</a></strong></li>

      <li><strong><a href="DataUser.jsp">Data User </a></strong></li>

      <li><strong><a href="CloudServer.jsp">Cloud Server </a></strong></li>

      <li><strong><a href="EndUser.jsp">End User </a></strong></li>

    </ul>

  </div>

</div>

</div>

<div class="fbg"></div>

<div class="footer">

</html>
```

CLOUD SERVER.JSP

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Cloud Server</title>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<link href="css/style.css" rel="stylesheet" type="text/css" />

<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />

<script type="text/javascript" src="js/cufon-yui.js"></script>

<script type="text/javascript" src="js/droid_sans_400-
droid_sans_700.font.js"></script>

<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>

<script type="text/javascript" src="js/script.js"></script>

<script type="text/javascript" src="js/coin-slider.min.js"></script>

<style type="text/css">

<!--

.style6 {color: #FF0000}

.style8 {color: #FF0000; font-weight: bold; }

-->

</style>

</head>

<body>
```

```
<div class="main">

<div class="header">

<div class="header_resize">

<div class="menu_nav">

<ul>

<li class="active"><a href="index.html"><span>Home Page</span></a></li>

<li><a href="DataUser.jsp"><span>Data User </span></a></li>

<li><a href="CloudServer.jsp"><span>Cloud Server </span></a></li>


</ul>

</div>

<div class="clr"></div>

<div class="logo">

<h1>&nbsp;</h1>

</div>

<div class="clr"></div>

<div class="slider">

<div id="coin-slider"> <a href="#"> </a> <a href="#"> </a> <a href="#"> </a> </div>

<div class="clr"></div>

</div>
```



```
<div class="clr"></div>

</div>

</div>

<div class="content">

<div class="content_resize">

<div class="mainbar">

<div class="article">

<h2><span>WELCOME TO CLOUD SERVER LOGIN </span></h2>

<p class="infopost">&nbsp; <a href="#" class="com"></a></p>

<div class="clr"></div>

<div class="img"></div>

<div class="post_content">

<form action="CAuthentication.jsp" method="post" id="leavereply">

<ol>

<li class="style8">

<label for="name">Name (required)</label>

<label for="name"></label>

<input id="name" name="userid" class="text" />

</li>

<li> <span class="style8">
```

```

        <label for="email">Password (required)</label>

    </span>

    <label for="email"></label>

    <input type="password" id="pass" name="pass" class="text" />

</li><li></li>

<li>

    <input name="imageField" type="submit" class="LOGIN"
id="imageField" value="Login" />

</li>

<li><br />

</li>

</ol>

</form>

<p class="spec"><a href="#" class="rm">..</a></p>

</div>

<div class="clr"></div>

</div>

</div>

<div class="sidebar">

<div class="searchform">

    <form id="formsearch" name="formsearch" method="post" action="#">

    <span>

```

```
<input name="editbox_search" class="editbox_search" id="editbox_search"
maxlength="80" value="Search our ste:" type="text" />

</span>

<input name="button_search" src="images/search.gif" class="button_search"
type="image" />

</form>

</div>

<div class="clr"></div>

<div class="gadget">

<h2 class="star"><span>Home </span> Menu</h2>

<div class="clr"></div>

<ul class="sb_menu">

<li><a href="index.html">Home</a></li>

<li><a href="DataUser.jsp">Data User </a></li>

<li><a href="

</div>

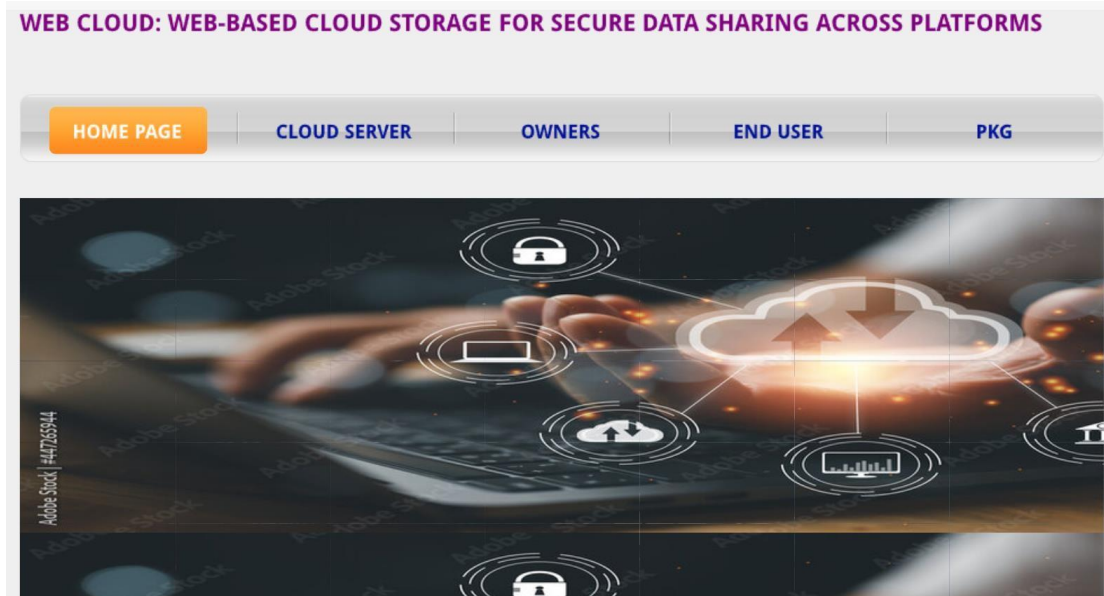
</div>

<div class="fbg"></div>

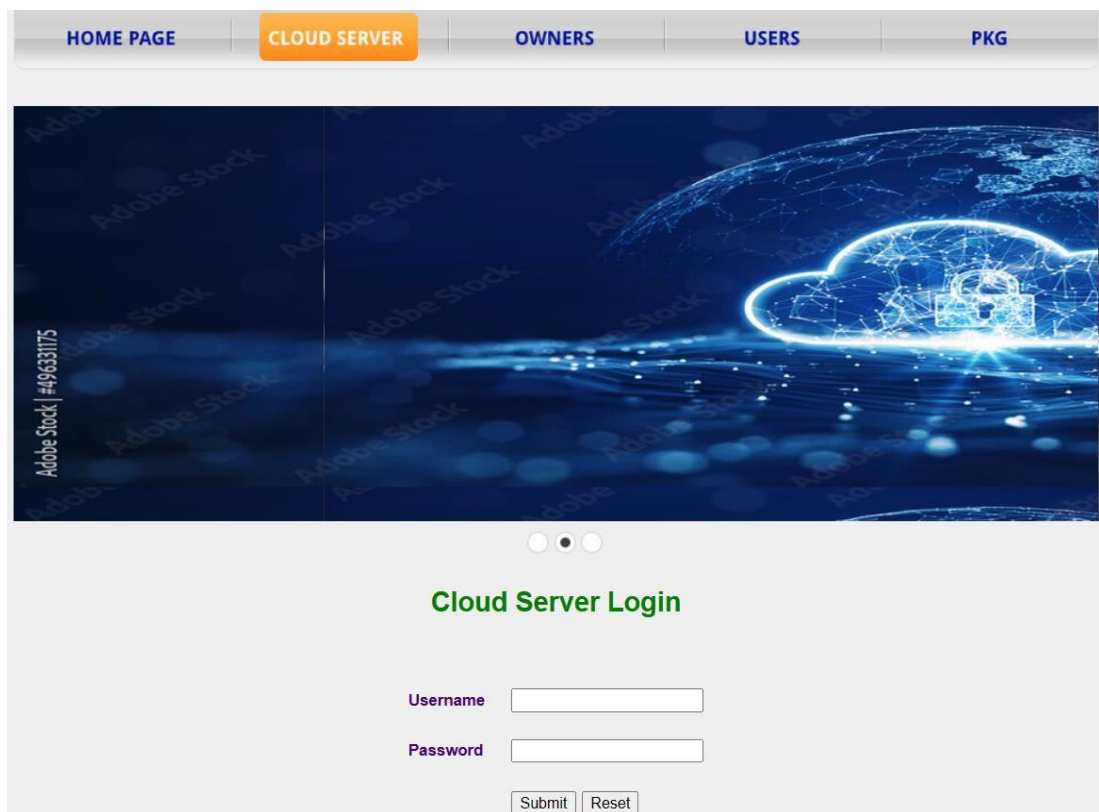
<div class="footer">

</html>
```

11. OUTPUT SCREENS



Screen 11.1. Home Page and Login Page



Screen 11.2. Cloud Server Login

HOME PAGE ADMIN OWNERS USERS PKG

Data Owner Login

Ownername

Password

[Home](#)

[New Registration](#)

Screen 11.3. Data Owner Login

Registration

Name

Username

Password

Email

Mobile

Address

DOB

Gender

Ownership type

Image No file chosen

Owner Key Update

Screen 11.4. Data Owner Registration

HOME PAGE ADMIN OWNERS **USERS** PKG

Adobe Stock | #49633175

User Login

Username

Password

[Home](#)

[New Registration](#)

Screen 11.5. Data User Registration

HOME PAGE ADMIN OWNERS **USERS** PKG

Registration

Name

Username

Password

Email

Mobile

Address

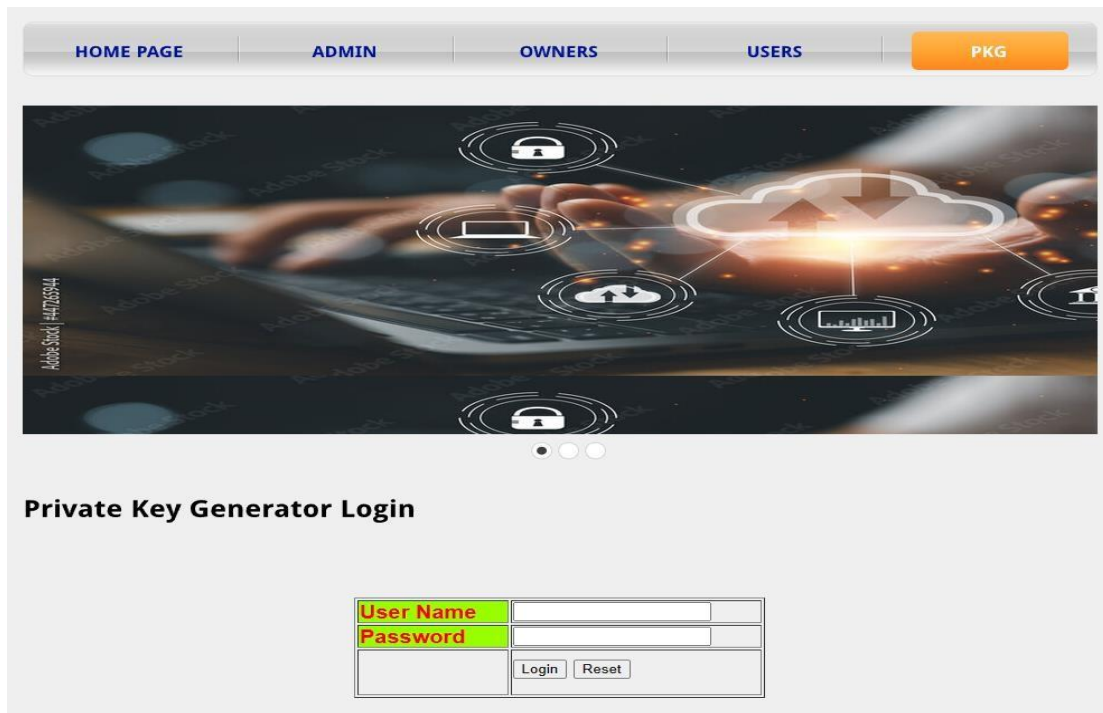
DOB

Gender ▼

Ownership type

Image No file chosen

Screen 11.6. Data Owner Registration

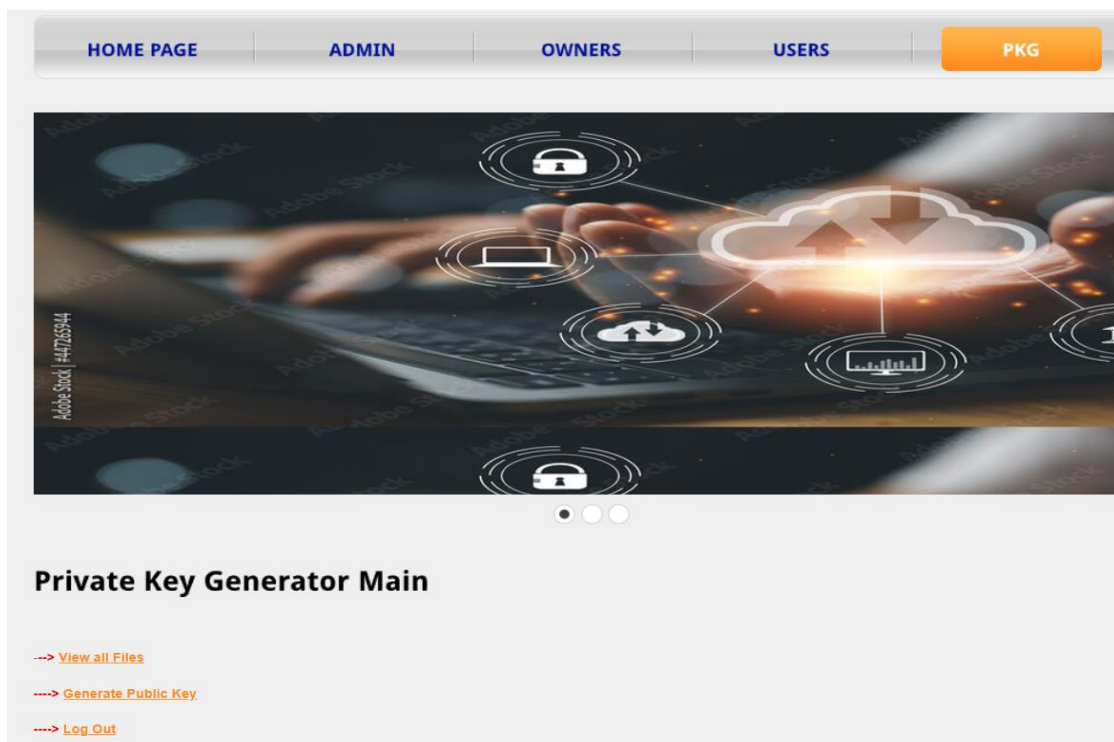


HOME PAGE ADMIN OWNERS USERS PKG

Private Key Generator Login

User Name	<input type="text"/>
Password	<input type="password"/>
	<input type="button" value="Login"/> <input type="button" value="Reset"/>

Screen 11.7. Private Key Generator Login



HOME PAGE ADMIN OWNERS USERS PKG

Private Key Generator Main

[--> View all Files](#)

[--> Generate Public Key](#)

[--> Log Out](#)

Screen 11.8. Private Key Generator Main

12. CONCLUSION AND FUTURE ENHANCEMENTS

12.1. CONCLUSION

Secure Data Sharing: Web Cloud has provided a reliable platform for secure data sharing across multiple platforms. Users can confidently store and share their data knowing that it is protected by robust security measures.

Cross-Platform Compatibility: The web-based nature of Web Cloud ensures compatibility with various devices and operating systems, making it accessible to a wide range of users.

User-Friendly Interface: With its intuitive user interface, Web Cloud simplifies the process of uploading, accessing, and sharing files, enhancing user experience and productivity.

12.2. FUTURE ENHANCEMENTS

- **Enhanced Security Features:** Continuously evolving security threats necessitate the implementation of advanced security features such as end-to-end encryption, two-factor authentication, and threat detection algorithms to further fortify data protection.

Mobile Application Development: Developing mobile applications for iOS and Android platforms can extend Web Cloud's accessibility, allowing users to access and manage their data on-the-go from their mobile devices.

13. REFERENCES

1. Roger S Pressman, “**Software Engineering – A Practitioner’s approach**” McGraw – Hill International Editions, Fifth Edition, 2001.
2. Henry F Korth, S. Sudharshan, “**Database System Concepts**” McGraw – Hill International Editions, Fourth Edition, 2002.
3. Herbert Schildt & Patrick Naughton, “**JSP**”, Tata McGraw- Hill International Editions, third edition, 1999.
4. James Jawroski, “**Java Script**”, Tata McGraw- Hill International Editions, third edition, 2000.
5. Karl Avedal “**JSP Architecture**”, Tata McGraw – Hill International Editions, Fourth Edition, 2002.